Análise Comparativa de Arquiteturas de Deep Learning (Stacked LSTM vs. 1D-CNN) no Reconhecimento de Atividades Humanas

Ramon Lima de Oliveira Tavares

2025-08-12

1 Introdução

O monitoramento automático de atividades humanas (HAR - *Human Activity Recognition*) consolidou-se como vetor de inovação na saúde digital (Lara & Labrador, 2013). Sensores inerciais miniaturizados geram fluxos de dados contínuos que, processados por algoritmos inteligentes, permitem inferir comportamentos complexos.

A problemática central reside na natureza estocástica e ruidosa das séries temporais do movimento humano. Enquanto métodos clássicos dependem de *feature engineering* manual, o Aprendizado Profundo (*Deep Learning*) aprende representações hierárquicas diretamente dos dados brutos (LeCun et al., 2015).

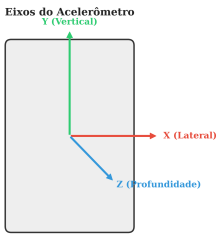
Este trabalho propõe um estudo comparativo rigoroso entre **Redes Neurais Convolucionais (1D-CNN)**, focadas em padrões locais (Kiranyaz et al., 2021), e **Redes Recorrentes (Stacked LSTM)**, focadas em dependências temporais (Hochreiter & Schmidhuber, 1997). O objetivo é avaliar a eficácia preditiva e a eficiência paramétrica de cada topologia no *dataset* UCI HAR.

2 Fundamentos Teóricos e Dados

2.1 Caracterização do Dataset UCI HAR

O estudo utiliza dados de 30 voluntários (19-48 anos) realizando Atividades de Vida Diária (ADL) com um *smartphone* na cintura, amostrado a 50Hz (Anguita et al., 2013).

Figura 1: Sistema de Coordenadas do Smartphone

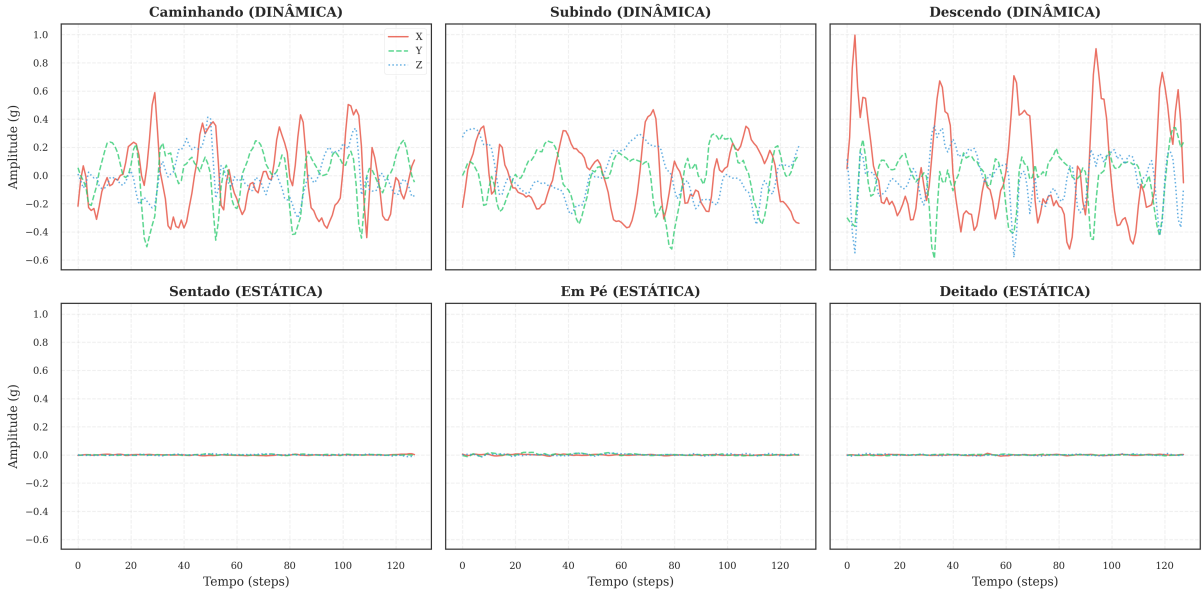


Fonte: Adaptado de Anguita et al. (2013).

2.2 Análise Visual dos Sinais Temporais

A complexidade do problema é evidenciada na visualização dos dados brutos. A Figura 2 demonstra a diferença estocástica entre atividades **Dinâmicas** (alta variância, picos agudos) e **Estáticas** (baixa amplitude, ruído constante).

Figura 2: Comparativo de Sinais Brutos (Janela 2.56s)



Legenda: Sinais normalizados de Aceleração e Giroscópio. Note a estacionariedade nas classes "Sentado", "Em Pé" e "Deitado". Fonte: Elaborada pelo autor.

2.3 Estrutura Algébrica e Volumetria

2.3.1 1. Definição de uma Amostra Única (Janela)

Cada instância de dado ($X^{(i)}$) é uma matriz densa representando 2,56 segundos de movimento contínuo.

$$X^{(i)} = \begin{bmatrix} \text{BodyAccX}_1 & \text{BodyAccY}_1 & \dots & \text{TotalAccZ}_1 \\ \text{BodyAccX}_2 & \text{BodyAccY}_2 & \dots & \text{TotalAccZ}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \text{BodyAccX}_{128} & \text{BodyAccY}_{128} & \dots & \text{TotalAccZ}_{128} \end{bmatrix} \in \mathbb{R}^{128 \times 9}$$

2.3.2 2. Volumetria Total dos Dados

O conjunto total foi particionado estaticamente para garantir reprodutibilidade. A dimensão massiva dos dados brutos processados é detalhada abaixo:

Tabela 1: Distribuição e Dimensão dos Dados

Conjunto	Quantidade de Amostras (N)	Dimensão por Amostra	Total de Pontos de Dados
Treinamento (70%)	7.352 janelas	128×9	8.469.504 valores
Teste (30%)	2.947 janelas	128×9	3.394.944 valores
TOTAL	**10.299 janelas**	-	**$\approx$ 11.8 Milhões de valores**

Fonte: Elaborada pelo autor.

2.4 Pré-processamento: Codificação One-Hot

Para viabilizar o treinamento supervisionado, os rótulos categóricos foram convertidos via **One-Hot Encoding**.

Tabela 1: Mapeamento de Classes e Codificação One-Hot

ID Original	Atividade	Índice Python	Vetor One-Hot (Target)
1	**Caminhando**	0	`[1, 0, 0, 0, 0, 0]`
2	**Subindo Escadas**	1	`[0, 1, 0, 0, 0, 0]`
3	**Descendo Escadas**	2	`[0, 0, 1, 0, 0, 0]`
4	**Sentado**	3	`[0, 0, 0, 1, 0, 0]`
5	**Em Pé**	4	`[0, 0, 0, 0, 1, 0]`
6	**Deitado**	5	`[0, 0, 0, 0, 0, 1]`

Fonte: Elaborada pelo autor.

2.5 Classificação Probabilística (Softmax)

A camada de saída utiliza a função **Softmax** para transformar os *logits* z em probabilidades normalizadas. A probabilidade $P(y = i)$ da amostra pertencer à classe i é dada por:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Onde: * z_i: Logit (saída linear) do neurônio correspondente à classe i. * K: Número total de classes ($K = 6$ neste estudo). * e: Constante de Euler (base do logaritmo natural). * $\sum_{j=1}^{K} e^{z_j}$: Fator de normalização que garante que a soma das probabilidades seja 1.

A predição final é a classe com maior probabilidade: $\hat{y} = \text{argmax}(\sigma(z))$ (Goodfellow et al., 2016).

3 Modelagem Computacional Avançada

A implementação utilizou Keras/TensorFlow. Abaixo, detalha-se a álgebra linear exata que ocorre dentro das camadas.

3.1 Arquitetura A: Stacked LSTM (Micro-dinâmica)

A rede processa a sequência temporal passo a passo ($t = 1 \dots 128$).

1. O Fatiamento (x_t): Embora a entrada seja uma matriz 128×9, a LSTM consome uma linha por vez. Para a multiplicação matricial correta, transpõe-se a linha para um vetor coluna:

$$x_t = (X_{t,:})^T \in \mathbb{R}^{9 \times 1}$$

2. A Fusão de Contexto (v_{in}): No passo t, a rede concatena a memória anterior (h_{t-1}) com a entrada atual (x_t):

$$v_{in} = \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} \in \mathbb{R}^{137 \times 1} \quad \text{(onde } 128 + 9 = 137\text{)}$$

3. Parâmetros e Portões: As matrizes de pesos (W) devem projetar esse vetor de 137 de volta para 128 (tamanho da célula).

$$W \in \mathbb{R}^{128 \times 137}$$

Isso justifica os **70.656 parâmetros** da primeira camada (4 gates $\times$ [$128 \times 137 + 128$ bias]).

Tabela 2: Resumo da Arquitetura Stacked LSTM

Camada	Configuração	Output Shape	Parâmetros
LSTM 1	128 units, return_seq=True	(None, 128, 128)	70.656
Dropout 1	Rate = 0.3	(None, 128, 128)	0
LSTM 2	64 units, return_seq=False	(None, 64)	49.408
Dense	64 units, ReLU	(None, 64)	4.160
Output	6 units, Softmax	(None, 6)	390
Total			**124.870**

3.2 Arquitetura B: Pure 1D-CNN (Convolução Multivariada)

A CNN aplica filtros que deslizam no tempo, mas integram espacialmente todos os sensores.

1. Anatomia do Kernel (W): O filtro ($K = 3$) não é um vetor, mas uma matriz que cobre **todas** as 9 variáveis de entrada simultaneamente.

$$W_{kernel} \in \mathbb{R}^{3\times 9}$$

2. Operação de Convolução: A cada passo de deslizamento, o kernel computa o produto escalar de todas as 9 variáveis na janela temporal de 3 passos. Como utilizamos **64 filtros** na primeira camada, temos 64 matrizes W distintas aprendendo padrões diferentes.

Tabela 3: Resumo da Arquitetura 1D-CNN

Camada	Configuração	Output Shape	Parâmetros
Conv1D 1	64 filtros, k=3	(None, 126, 64)	1.792
Conv1D 2	128 filtros, k=3	(None, 61, 128)	24.704
Conv1D 3	256 filtros, k=3	(None, 28, 256)	98.560
GAP	Global Average Pooling	(None, 256)	0
Output	6 units, Softmax	(None, 6)	1.542
Total	(Incluindo Dense Interm.)		**143.686**

4 Análise de Resultados

4.1 Dinâmica de Convergência e Teste

A CNN convergiu mais rapidamente, demonstrando maior facilidade em otimizar gradientes em comparação à recorrência temporal (BPTT) da LSTM. Nos dados de teste ($N = 2.947$), a **1D-CNN** superou a LSTM, atingindo **93% de acurácia global** contra 90%.

Tabela 4: Comparativo de Métricas por Classe (F1-Score)

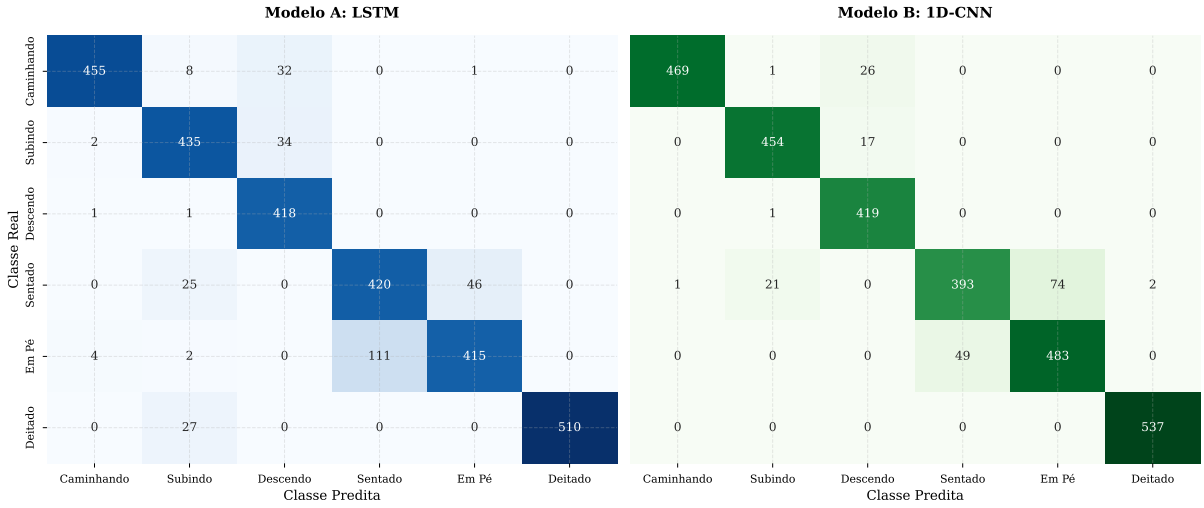
Atividade	Stacked LSTM	Pure 1D-CNN	Δ Desempenho
Caminhando	0.95	**0.97**	+2%
Subindo Escadas	0.90	**0.96**	+6%
Descendo Escadas	0.92	**0.95**	+3%
Sentado	0.82	**0.84**	+2%
Em Pé	0.84	**0.89**	+5%
Deitado	0.97	**1.00**	+3%
Média Global	**0.90**	**0.93**	**+3%**

Fonte: Elaborada pelo autor.

4.2 Limitações Físicas (Análise de Erros)

A Figura 3 expõe a principal fronteira de erro: a distinção **Sentado vs. Em Pé**.

Figura 3: Matrizes de Confusão Normalizadas



Legenda: Esquerda: LSTM; Direita: CNN. A diagonal principal indica acertos. Fonte: Elaborada pelo autor.

Em repouso estático, a magnitude da aceleração é dominada pela gravidade ($g \approx 9.8 m/s^2$). Sem variação significativa no giroscópio e sem um sensor de altitude, os vetores de "Sentado" e "Em Pé" são geometricamente quase idênticos (tronco vertical). A CNN lidou melhor com isso, sugerindo que ela identificou micro-padrões de vibração postural que a LSTM ignorou.

5 Conclusão

Este estudo confirmou que a extração automática de características via Deep Learning é eficaz para HAR. A arquitetura **1D-CNN** provou-se superior à **Stacked LSTM**, não apenas em acurácia (93%), mas em eficiência de treinamento e generalização para a classe "Deitado" (100%). Matematicamente, a convolução mostrou-se mais robusta para capturar a morfologia do sinal em janelas curtas do que a recorrência pura.

6 Referências

ANGUITA, D. et al. A Public Domain Dataset for Human Activity Recognition Using Smartphones. *ESANN*, 2013.

GERS, F. A.; SCHMIDHUBER, J.; CUMMINGS, F. Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10), 2000.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. MIT Press, 2016.

HE, K. et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *ICCV*, 2015.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. *Neural Computation*, 9(8), 1997.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

KIRANYAZ, S. et al. 1D Convolutional Neural Networks and Applications: A Survey. *MSSP*, 151, 2021.

LARA, O. D.; LABRADOR, M. A. A survey on human activity recognition using wearable sensors. *IEEE communications surveys & tutorials*, 2013.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, 521, 2015.

SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1), 2014.